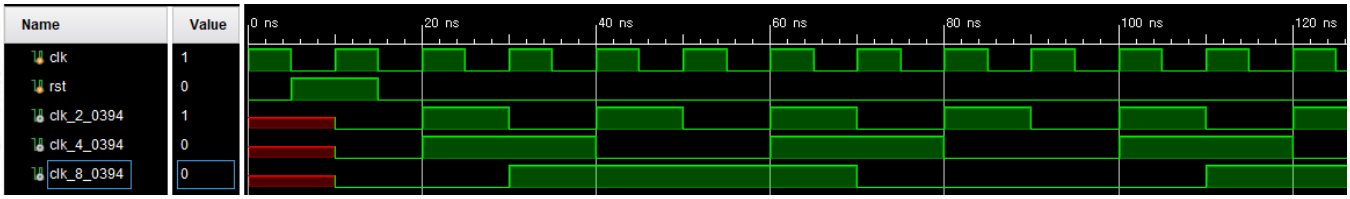
설계과제 HW4 20160394 임효상

1) 2^N divider




```
module divide_by_2(
input clk, input reset,
output reg clk_2, clk_4, clk_8);
reg [2:0] count;

always @(posedge clk) begin
    if(reset) begin
        count <= 4'b0;
        clk_2 <= 1'b0;
        clk_4 <= 1'b0;
        clk_8 <= 1'b0;
    end
    else begin
        count = count + 1;

        clk_2 <= ~clk_2;

        if(count[0]) begin
            clk_4 <= ~clk_4;
        end
        else if(count[1]) begin
            clk_8 <= ~clk_8;
        end
    end
end

endmodule
```

```
module tb_clk_div();

reg clk, rst;
wire clk_2_0394, clk_4_0394, clk_8_0394;

initial begin
    clk <= 1;
    rst <= 0;
    #5
    rst <= 1;
    #10
    rst <= 0;
end

always #5 clk <= ~clk;

divide_by_2 clk_24816(clk, rst, clk_2_0394, clk_4_0394, clk_8_0394);

endmodule
```

Counter를 사용해 2^N divider를 구현했다.

[3]크기의 count 배열을 이용한다.

배열의 [0]이 바뀔때 마다 clk_4가 변화한다.

이를 통해 2^2 divider 효과를 볼 수있다.

배열의 [1]이 바뀔때마다 clk_8이 변화한다.

이를 통해 2^3 divider 효과를 볼수있다.

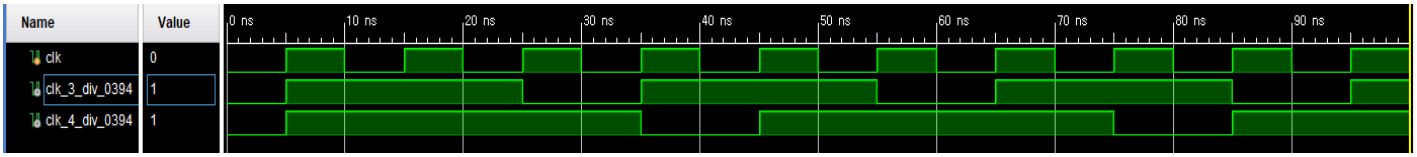
매 posedge 마다 clk_2가 변화한다.

이를 통해 2^1 divider 효과를 볼 수 있다.

이와 같은 방법으로 배열크기를 확장하면 2^N divider를 만들 수 있다.

하지만 2^3 divider 부터 clock의 posedge 지점이 코드 구조 상 달라진다.

2) N Divider

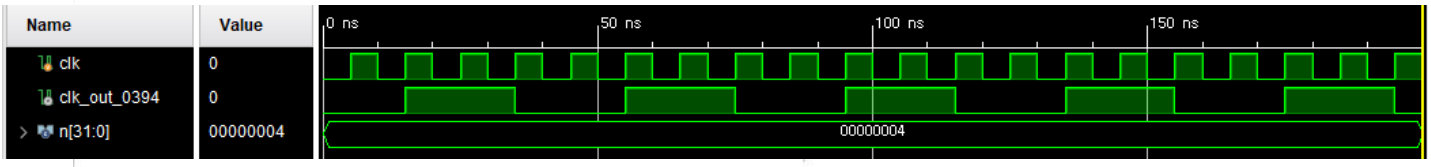


N divider는 입력 주파수를 1/N 배 되는 출력 주파수를 만든다.

3 divider는 기존 입력되는 클럭 주기의 3배가 되도록 출력을 내보낸다. N이 홀수이므로 50% duty 비로는 나누지 못한다.

4 divider는 기존 입력되는 클럭 주기의 4배가 되도록 출력을 내보낸다. N이 짝수이므로 50% duty 비로 나눌수 있지만 코드 상 그렇게 하지 않았다.

3) 파라미터 값을 받는 N divider




```
reg clk;
parameter n = 4;
wire clk_out_0394;
                0

initial clk = 0;
always #5 clk = ~clk;

clk_Ndiv clk_ndiv(clk,n,clk_out_0394);
```

클럭 100MHz, n=4로 설정했다.
왼쪽은 test bench 코드이다.
n=4로 1/4 주파수로 클럭을 나눈것을 볼 수 있다.